PR #52112 完整报告

vllm-project/vllm

[Bugfix][ROCm] Fix a few int4/int8 quantization errors

合并时间：2026-08-18 16:31

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52112

执行摘要

- 一句话：修复 ROCm 平台 int4/int8 量化 MoE 回归，扩 TRITON 后端能力
- 推荐动作：值得精读，尤其推荐关注 int_wna16.py 中 ZP 布局转换的逐步推导注释——这是理解 fused_moe_kernel_gptq_awq 数据布局的绝佳材料。适合对 MoE 量化、TP 切分与 Triton kernel 数据契约感兴趣的读者。建议后续补充针对 ZP 布局转换和 packed_factor 校验的单元测试，并考虑在文档中明确非对称 WNA16 的 TP 约束。

功能与动机

PR #44120 将 MoeWNA16Method 迁移到新的 MK oracle 方案后，在 ROCm 平台引入了若干量化回归：一是 compressed-tensors 路径对非对称量化直接断言拒绝，导致 MiniMax-M3-AWQ-INT4 这类带 zero point 的 AWQ 模型无法走 TRITON MoE 后端；二是 int8 量化被错误地限定为 group_size==-1（无 group 量化），QuantTrio/Qwen3-235B-A22B-GPTQ-Int8 这类带 group size 的 GPTQ-Int8 模型直接断言失败；三是 TRITON MoE 后端不支持 MiniMax-M3 所需的 SWIGLUOAI_UNINTERLEAVE 激活。PR body 明确列出两个目标模型的 serve 测试命令并验证输出正确。

实现拆解

本 PR 分四步修复回归，全部落在 vllm/model_executor/layers 下的量化与 MoE 路径：

1. 放开对称量化限制并增加 packed_factor 校验（quantization/compressed_tensors/compressed_tensors_moe/compressed_tensors_moe_wna16.py） - 删除 assert self.symmetric，使非对称（asymmetric）WNA16 可以进入非 Marlin 后端（含 TRITON）分支。 - 在 create_weights 中新增非对称量化下的整除性校验：2 * intermediate_size_per_partition 与 hidden_size 必须能被 packed_factor（int4 时为 8）整除，因为 int4 的 zero point 以 int32 打包存储，TP 切分时每个 rank 必须持有整数个 int32 元素，否则 ZP 会被从字节中间截断导致解包错误。 - 调整 process_weights_after_loading 中 ZP 替换条件：EMULATION 后端会把 ZP 烘焙进反量化后的 bf16 权重，返回的 w13_qzeros/w2_qzeros 为 None，因此需显式排除 EMULATION，避免无谓的断言失败。
2. 新增 TRITON 后端 ZP 布局转换（fused_moe/oracle/int_wna16.py） - 在 convert_to_wna16_moe_kernel_format 的 TRITON 分支中，对来自 compressed-tensors/auto_gptq 的 K-first int32 零点张量做布局变换：(E, K//gs, N//8) int32 依次经 transpose、view(uint8)、reshape、permute 转为 (E, N//2, K//gs) uint8，

使每字节携带 2 个 int4 ZP，匹配 fused_moe_kernel_gptq_awq 的索引约定 (offs_bn // 2) * stride_bzn + offs_k_group * stride_bzk。

3. 扩展 TRITON WNA16 专家后端的能力声明（fused_moe/experts/triton_moe.py） - 导入 kInt4StaticAsym、kInt4Static32Asym 并加入 TritonWNA16Experts._supports_quant_scheme 的支持列表，使 oracle 能对该后端放行非对称 int4 方案。 - 将 MoEActivation.SWIGLUOAI_UNINTERLEAVE 加入 _supports_activation，并同步修正注释，说明该激活经由 apply_moe_activation() 路由到 torch.ops._C.silu_and_mul_with_clamp。
4. 移除 int8 的 group_size 限制（quantization/moe_wna16.py） - 删除 assert group_size == -1，使 int8 量化支持 group_size != -1（即 group-wise int8 量化，如 Qwen3-235B-A22B-GPTQ-Int8），为这类模型构建出正确的 QuantKey(kInt8StaticGroupScale, ...)。
5. 测试与配套：PR 未新增自动化测试，仅以两个真实模型（MiniMax-M3-AWQ-INT4、Qwen3-235B-A22B-GPTQ-Int8）的端到端 serve 命令作为测试计划；CI 由 tjtanaa 触发两次并通过。

关键文件：

- vllm/model_executor/layers/fused_moe/oracle/int_wna16.py（模块 量化编排；类别 source；类型 data-contract；符号 convert_to_wna16_moe_kernel_format）：核心算法改动：为 TRITON WNA16 后端补齐 compressed-tensors 来源 zero point 的布局转换（K-first int32 → N-first uint8），这是解锁非对称 AWQ/GPTQ 模型的关键，转换链路含详细的逐步推导注释。
- vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors_moe/compressed_tensors_moe_wna16.py（模块 量化层；类别 source；类型 data-contract；符号 CompressedTensorsMoeWNA16Method.create_weights, CompressedTensorsMoeWNA16Method.process_weights_after_loading）：移除对称量化断言并增加 packed_factor 整除性校验，同时修正 EMULATION 后端的 ZP 处理；这是让非对称 WNA16 能进入 TRITON 后端的入口级变更。
- vllm/model_executor/layers/fused_moe/experts/triton_moe.py（模块 MoE 后端；类别 source；类型 data-contract；符号 TritonWNA16Experts._supports_quant_scheme, TritonWNA16Experts._supports_activation）：TRITON WNA16 专家后端的方案与激活能力声明扩展：新增非对称 int4 量化键和 SWIGLUOAI_UNINTERLEAVE 激活支持，是 oracle 放行路径与上层模型接入的桥梁。
- vllm/model_executor/layers/quantization/moe_wna16.py（模块 量化层；类别 source；类型 data-contract；符号 MoeWNA16Method.init）：移除 int8 量化必须 group_size==-1 的错误断言，使 group-wise int8（如 Qwen3-235B-A22B-GPTQ-Int8）能正确构建 QuantKey 并走 TRITON 后端。

关键符号：convert_to_wna16_moe_kernel_format, CompressedTensorsMoeWNA16Method.create_weights, CompressedTensorsMoeWNA16Method.process_weights_after_loading, TritonWNA16Experts._supports_quant_scheme, TritonWNA16Experts._supports_activation, MoeWNA16Method.init

关键源码片段

vllm/model_executor/layers/fused_moe/oracle/int_wna16.py

核心算法改动：为 TRITON WNA16 后端补齐 compressed-tensors 来源 zero point 的布局转换（K-first int32 → N-first uint8），这是解锁非对称 AWQ/GPTQ 模型的关键，转换链路含详细的逐步推导注释。

```
# 文件：vllm/model_executor/layers/fused_moe/oracle/int_wna16.py
# 位置：convert_to_wna16_moe_kernel_format() 的 TRITON 分支内
#
# compressed-tensors checkpoints 的 zero point 是 K-first int32，每个元素
# 打包 8 个 int4 ZP，形状为 (E, K//gs, N//8)；而 fused_moe_kernel_gptq_awq
# 期望 N-first uint8、每字节携带 2 个 int4 ZP，形状为 (E, N//2, K//gs)，
# 索引方式为 (offs_bn // 2) * stride_bzn + offs_k_group * stride_bzk。
if w13_qzeros is not None:
    E13, Kg13, Np13 = w13_qzeros.shape
    # 转换链路（每步都依赖 view/reshape 的内存连续性）：
    # (E, K//gs, N//8) int32
    # → transpose(1,2) → (E, N//8, K//gs) int32
    # → view(uint8) → (E, N//8, K//gs*4) 每个 int32 拆成 4 字节
    # → reshape(..., 4) → (E, N//8, K//gs, 4) 隔离出字节下标
    # → permute(0,1,3,2) → (E, N//8, 4, K//gs) 字节下标挪到 K 之前
    # → reshape → (E, N//2, K//gs) 得到 kernel 期望布局
    # 转换后元素 [e, offs_bn//2, k_group] 即为同时持有输出通道
    # offs_bn 与 offs_bn+1 两个 int4 ZP 的 uint8 字节。
    w13_qzeros = (
        w13_qzeros.transpose(1, 2)
        .contiguous()
        .view(torch.uint8)
        .reshape(E13, Np13, Kg13, 4)
        .permute(0, 1, 3, 2)
        .reshape(E13, Np13 * 4, Kg13)
        .contiguous()
    )
if w2_qzeros is not None:
    E2, Kg2, Np2 = w2_qzeros.shape
    # 与 w13 完全相同的转换逻辑，作用于 MoE 的 w2 权重（输出投影）。
    w2_qzeros = (
        w2_qzeros.transpose(1, 2)
        .contiguous()
        .view(torch.uint8)
        .reshape(E2, Np2, Kg2, 4)
        .permute(0, 1, 3, 2)
        .reshape(E2, Np2 * 4, Kg2)
        .contiguous()
    )
```

vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors_moe/compressed_tensors_moe_wna16.py

移除对称量化断言并增加 packed_factor 整除性校验，同时修正 EMULATION 后端的 ZP 处理；这是让非对称 WNA16 能进入 TRITON 后端的入口级变更。

```
# 文件：vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors_moe/compressed_tensors_moe_wna16.py
# 位置：CompressedTensorsMoeWNA16Method.create_weights() 计算 num_groups 之后
if not self.symmetric:
    # 非对称 int4 量化下，8 个 int4 zero point 打包进 1 个 int32 元素；
    # TP 切分时每个 rank 必须拿到整数个 int32，否则 packed ZP 会被从
    # int32 中间截断，解包结果错乱，因此提前做整除性校验并给出明确报错。
    w13_n = 2 * intermediate_size_per_partition # gate+up 输出通道数
    if w13_n % self.packed_factor != 0:
        raise ValueError(
            f"CompressedTensors WNA16 MoE: gate+up output channels per "
            f"TP rank (2 * intermediate_size_per_partition = {w13_n}) "
            f"must be divisible by packed_factor ({self.packed_factor}). "
            f"Use a TP size where 2 * intermediate_size is divisible by "
            f"{self.packed_factor}."
        )
    if hidden_size % self.packed_factor != 0:
        raise ValueError(
            f"CompressedTensors WNA16 MoE: hidden_size ({hidden_size}) "
            f"must be divisible by packed_factor ({self.packed_factor}) "
            f"for correct ZP unpacking."
        )

# 位置：process_weights_after_loading() 中替换权重参数的阶段
# CPU fused_experts_cpu 即使对称量化也需要 zero point；而 EMULATION
# 后端在 convert 阶段就把 ZP 烘焙进反量化后的 bf16 权重，返回的
# w13_qzeros/w2_qzeros 为 None，因此必须显式排除，避免误断言。
if (
    not self.symmetric or self.wna16_backend == WNA16MoEBackend.CPU
) and self.wna16_backend != WNA16MoEBackend.EMULATION:
    assert w13_qzeros is not None and w2_qzeros is not None
    replace_parameter(layer, "w13_weight_zero_point", w13_qzeros)
    replace_parameter(layer, "w2_weight_zero_point", w2_qzeros)
```

评论区精华

本 PR 没有实质性的 review 技术讨论，代码评审流程简洁：

- claude[bot] 因 PR 来自 fork 自动跳过审查，提示维护者可用 @claude review 触发一次性审查。
- 维护者 tjtanaa 直接批准（APPROVED）并两次触发 Buildkite CI（#84194、#84345），说明改动在维护者视角风险可控、验证充分。

- 关联 Issue #44120 的讨论集中在迁移动机（将 MoeWNA16Method 迁移到 MK oracle 方案）与自动 GPTQ 权重转换修复，本 PR 属于该迁移的后续回归修复，未再展开设计争论。
- CI 触发与运行状态 (other): 两次 CI 均被触发，PR 最终由 tjtanaa 批准合并，未在评论中暴露失败信息。
- fork PR 的自动化审查策略 (other): 未触发额外审查，由人类维护者直接批准。

风险与影响

- 风险：
  1. ZP 布局转换正确性风险（fused_moe/oracle/int_wna16.py）：新增的 transpose/view/reshape/permute 链依赖对 fused_moe_kernel_gptq_awq 索引语义的精确理解，一旦 compressed-tensors 侧的打包约定或 kernel 侧索引方式变化，容易静默产出错误结果；本 PR 未附带单元测试覆盖此转换，回归风险较高。
  2. TP 配置兼容性（compressed_tensors_moe_wna16.py）：新增的 packed_factor 整除性校验会在某些 TP 规模下直接抛 ValueError 拒绝启动。这是显式失败而非静默错误，比原先更安全，但可能让部分既有非对称模型配置从“可运行但结果错误”变成“无法启动”，需在文档中说明 TP 选择约束。
  3. EMULATION 分支的 ZP 缺失（compressed_tensors_moe_wna16.py）：EMULATION 后端不再断言 ZP 非 None，若未来 EMULATION 实现改为要求在权重中保留 ZP，则该分支会静默丢掉 ZP 信息。
  4. int8 group-wise 支持面扩大（moe_wna16.py）：移除 group_size == -1 断言后，kInt8StaticGroupScale 会覆盖所有 group size，但 kernel 是否完整支持任意 group size（如非 128 的 group）未在本 PR 中验证，存在潜在未覆盖路径。
  5. 缺少自动化测试：两个手工 serve 命令依赖特定模型权重（体积大），难以进入常规 CI，后续重构容易再次打破这些路径。
- 影响：
  - 用户影响：ROCm 平台上 int4/int8 量化 MoE 模型（尤其 AWQ-INT4 与 GPTQ-Int8 带 group 或 zero point 的模型）从“无法启动 / 断言失败”恢复为可用，直接解锁 MiniMax-M3 与 Qwen3-235B 等模型在 AMD GPU 上的推理。
  - 系统影响：改动触及 MoeWNA16Method 公共选择路径与 TRITON MoE 专家后端的方案声明，属于核心量化数据契约变更；对 NVIDIA 平台影响有限（TRITON 后端跨 CUDA/ROCm 共用，但本 PR 主要解锁 ROCm 场景）。
  - 团队影响：为后续在 TRITON MoE 后端支持非对称 WNA16 提供了参考实现（ZP 布局转换），可复用于其他模型接入；同时暴露了 #44120 迁移缺乏回归测试覆盖的问题。
  - 风险标记：缺少自动化测试覆盖，ZP 布局转换依赖 kernel 索引语义，新增 TP 配置拒绝逻辑，int8 group 支持面未验证

关联脉络

- PR #44120 [MoE Refactor] Migrate MoeWNA16Method quantization method over to using the new MK oracle scheme.: 本 PR 明确声明是为修复 #44120 引入的回归而存在；#44120 将 MoeWNA16Method 迁移到 MK oracle 选择方案，并叠加了 Triton 权重转

换与 Marlin ZP 转换的改动，本 PR 是它的后续 bugfix。

- PR #52044 [Bugfix] Handle DeepseekV4ForCausalLM in benchmark_moe get_model_params: 同属 DeepSeek 系 MoE 量化 / 参数解析回归修复，反映近期 MoE 量化重构（#44120 路线）引入的多点回归，需要逐个修复。
- PR #52625 [ROCm] gaurd on_gfx1250 call with rocm platform: 同为 ROCm 平台量化工具链的守卫 / 兼容性修复，属于同一波 ROCm 量化路径问题收敛。